

ECS Configuration Change Request

Page 1 of

Page(s)

1. Originator Cherry Kenney	2. Log Date: 11/21/01	3. CCR #: 01-0890	4. Rev: —	5. Tel: 4177	6. Rm #: 3204D	7. Dept. DEV
8. CCR Title: TE_6A.04_DDIST.01 For All DAACs. Contains DDIST fix for Sev 5 NCR - DDIST Cannot Throttle FtpPush Requests and SEv 2 NCR StagingDiskServer Not able to reuse same rpid in Create						
9. Originator Signature/Date Cherry Kenney /s/ 11/21/01			10. Class II	11. Type: CCR	12. Need Date: 21NOV01	
13. Office Manager Signature/Date Arthur Cohen /s/ 11/21/01			14. Category of Change: Initial ECS Baseline Doc.		15. Priority: (If "Emergency" fill in Block 27). Emergency	
16. Documentation/Drawings Impacted:			17. Schedule Impact:		18. CI(s) Affected: DDIST	
19. Release Affected by this Change: 6A		20. Date due to Customer: 21NOV01		21. Estimated Cost: None - Under 100K		
22. Source Reference: <input checked="" type="checkbox"/> NCR (attach) <input type="checkbox"/> Action Item <input type="checkbox"/> Tech Ref. <input type="checkbox"/> GSFC <input type="checkbox"/> Other: 32026, 32383						
23. Problem: (use additional Sheets if necessary) 32026 - DDIST Cannot Throttle FtpPush Requests 32383 - StagingDiskServer Not able to reuse same rpid in Create						
24. Proposed Solution: (use additional sheets if necessary) Note: This Test Executable (TE) changes the way that Data Distribution (DDIST) queues and schedules distribution requests. Please ensure that the appropriate System Engineering, DBA, Ops Controller, and Distribution staff read this document before promoting this TE into the OPS mode. Contact Miguel Zuniga (mzuniga@eos.hitc.com) or Steve Fox (sfox@eos.hitc.com) if you have questions about this capability See Page two for additional notes:						
25. Alternate Solution: (use additional sheets if necessary) Wait until next s/w release						
26. Consequences if Change(s) are not approved: (use additional sheets if necessary) Delays in getting fixes to the field, DAACs must continue to work around problems						
27. Justification for Emergency (If Block 15 is "Emergency"): NCR fixes are needed at the DAACs to support the 6A.04 baseline in OPS mode						
28. Site(s) Affected: <input type="checkbox"/> EDF <input checked="" type="checkbox"/> PVC <input checked="" type="checkbox"/> VATC <input checked="" type="checkbox"/> EDC <input checked="" type="checkbox"/> GSFC <input checked="" type="checkbox"/> LaRC <input checked="" type="checkbox"/> NSIDC <input type="checkbox"/> SMC <input type="checkbox"/> AK <input type="checkbox"/> JPL <input type="checkbox"/> EOC <input type="checkbox"/> IDG Test Cell <input type="checkbox"/> Other						
29. Board Comments:			30. Work Assigned To:		31. CCR Closed Date:	
32. EDF/SCDV CCB Chair (Sign/Date): Byron V. Peters /s/ 11/21/01		Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS ERB				
33. M&O CCB Chair (Sign/Date): Pamela Johnson /s/ 11/20/01		Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS Fwd/ECS				
34. ECS CCB Chair (Sign/Date):		Disposition: Approved App/Com. Disapproved Withdraw Fwd/ESDIS Fwd/ESDIS				

CCR #: **Rev:** **Originator:** Cherry Kenney

Telephone: 301-883-4177 **Office:** 3204D

Title of Change: TE_6A.04_DDIST.01 For All DAACs. Contains DDIST fix for Sev 5 NCR - DDIST Cannot Throttle FtpPush Requests

Clearcase:

Please roll up the following packages from the 6A04NOFW baseline in one SUN tar files and one IRIX65 tar files.

Sun

.EcDsDdDISTSRVR.pkg
.EcDsDdDdistGUI.pkg

SGL:

.EcDsStDatabase.pkg

PLEASE NAME TARFILE TE_6A.04_DDIST.01

Proposed Solution continued :

Note: This Test Executable (TE) adds one column (ThreadPoolId) to the DsDdRequest table in the DDIST database. Please ensure that this does not impact any DAAC-Unique Extensions before promoting this TE into the OPS mode.

Please Read Thread Pool Document Before Installing Test Executable

1. Introduction

Currently, DDIST supports five thread pools that are mapped to the system priority levels (XPRESS, VHIGH, HIGH, NORMAL, and LOW). The DAAC can configure the number of threads associated with each priority level and DDIST will allocate requests to these thread pools based on request priority. In addition, DDIST will allocate a thread from a lower priority pool to a request if all threads at its assigned priority are in use.

As documented in NCR 32026, this approach provides limited control over preventing slow requests from a single user from blocking other user requests. This problem can occur when the throughput of a user network connection is insufficient to keep up with the volume of data being distributed to the user. This can be a temporary condition due to network/destination host problems or it can be a permanent condition if the volume of data being distributed via subscription is too large. The bottleneck occurs when multiple distribution requests for this user begin to backup and consume all available distribution threads at the assigned and lower priority levels. At this point any other distribution requests at the same and lower priority levels become blocked until the requests going over the slow connection are worked off. With the current mechanism, the only way to potentially avoid the problem is to map each user to a different priority level. Since there are only five priority levels available, this isn't a very feasible solution.

This TE introduces a new DDIST resource management capability that enables the DAAC to define distribution thread pools tailored for their workload along with a set of rules for assigning distribution requests to thread pools. This approach not only provides greater flexibility to address the problem described in the NCR but also provides a mechanism for ensuring distribution resources are effectively allocated across users and media devices.

The following sections describe the thread pool operations concept and the specific procedures for creating, deleting, modifying, and trouble-shooting thread pools and assignment rules.

2 Thread Pool Operations Concept

DDIST has been enhanced to support a DAAC-configurable number of thread pools with each pool having a separate thread limit. These pools are defined in a new DDIST database table called DsDdThreadPools. Each row in this table defines a unique pool identifier, pool name, and the number of threads (thread limit) associated with the pool. Table 2-1 shows an example table.

Table 2-1 Example DsDdThreadPools Table

ThreadPoolId	ThreadPoolName	ThreadLimit
1	MODAPS	20
2	DLT_ORDERS	2
3	DEFAULT	30

This table defines three pools. The MODAPS pool has 20 threads and the DLT_ORDERS pool has 2 threads. It is mandatory that the DEFAULT pool be defined, as any requests that fail to match an assignment rule will be automatically allocated to the DEFAULT pool.

A second new DDIST table called DsDdAssignmentRules contains a set of DAAC-configurable rules for assigning a new request to a pool. These rules are based on request attributes. The attributes that can be used to determine thread pool assignment include: ECSUserId, Priority, EsdtType, and MediaType. Each row in this table will define an assignment rule. Table 2-2 shows an example table.

Table 2-2 Example DsDdAssignmentRules Table

SeqNum	EcsUserId	Priority	EsdtType	MediaType	ThreadPoolId
100	Robbie	ANY	ANY	FtpPush	1
200	ANY	ANY	ANY	DLT	2

For each new request, the rules are evaluated in order by SeqNum until a rule is found where all conditions evaluate to true, in which case the request is assigned to the pool specified in the ThreadPoolId column. A rule evaluates to true if the values of the request attributes (i.e., ECSUserId, Priority, EsdtType, and MediaType) match the values contained in the row. Note that a value of ANY will automatically evaluate to true for that attribute. So, in the example, any FtpPush request with an ECSUserId of Robbie will be allocated to the MODAPS thread pool and all DLT requests will be allocated to the DLT_ORDERS thread pool. Any requests that fail to match one of these two rules will be allocated to the DEFAULT thread pool.

When a request enters DDIST, a new stored procedure is executed that will assign the request to a particular thread pool based on the rules contained in DsDdAssignmentRules. Once all threads in a given thread pool have been assigned, new requests assigned to that pool will be put in a pending state until a thread is available. Requests will no longer be automatically assigned to threads in other pools if there are no available threads in their assigned pool. Pending requests for each pool will be activated in first-in-first-out order by request priority.

DAACs may adjust configurations by updating the DsDdThreadPools and DsDdAssignmentRules tables. Assignment rules may be added, deleted or updated at any time without warm-starting DDIST. Changes to assignment rules will take effect immediately and all new requests entering DDIST will be subject to the updated rules. The ThreadLimit attribute in the DsDdThreadPools table may be dynamically changed as well. The DDIST server reloads thread limits every 90 seconds so thread limit changes will take effect within 90 seconds after being entered. New thread pools can be added by inserting rows in the DsDdThreadPools table. However, they won't be used until the DDIST server is warm-started. A thread pool can be deleted as long as there are no rules in the DsDdAssignmentRules table that point to the thread pool and all requests that have been assigned to the thread pool have been completed and have migrated out of the DDIST database.

When DDIST is warm-started, all requests are reassigned to thread pools based on the current set of rules. If it is desired to reassign requests after they have been assigned to a thread pool, then the rules in the DsDdAssignmentRules table should be updated to effect the desired assignment and DDIST warm-started.

A thread pool GUI is not provided with the initial delivery of this capability. Instead, thread pool configuration changes are made by a DAAC DBA using the isql interface to update the DsDdThreadPools and DsDdAssignmentRules tables.

3 Rules for Creating and Modifying Thread Pools and Assignment Rules

Each thread pool is defined by a row in the DsDdThreadPools table. The following attributes must be specified for each row:

ThreadPoolId is the unique identifier for the thread pool. It is an integer whose value must be greater than zero. Each row in DsDdThreadPools must have a unique ThreadPoolId.

ThreadPoolName is the name of the thread pool. It is a string whose length is less than or equal to 24 characters. Each row in DsDdThreadPools must have a unique ThreadPoolName.

ThreadLimit is the number of threads available for processing requests assigned to this thread pool. It is an integer whose value is greater than or equal to zero. If the ThreadLimit for a given thread pool is zero, any requests that are assigned to the thread pool will remain in the pending state until the ThreadLimit is set to a value greater than zero. If the ThreadLimit for a given thread pool is updated from a non-zero value to zero, no new requests assigned to the thread pool will be activated. However, any currently active requests will be allowed to complete. The total of the thread limits for all thread pools must be less than the number of worker threads configured for DDIST. The default worker thread configuration for DDIST is 228 threads.

The DDIST server must be warm-started before it will recognize new thread pools. The correct procedure is:

Halt the DDIST server

Using isql, add new rows to the DsDdThreadPools table

Warm-start the DDIST server

If a new pool is added to DsDdThreadPools and new rules are added to DsDdAssignmentRules while the DDIST server is running and these new rules result in a request being assigned to the new pool then the request will become suspended with an DsEDdMissingPool error code. The request will be unable to be resumed until the DDIST server is warm-started.

A thread pool cannot be deleted if any rule in the DsDdAssignmentRules table references the thread pool or any request in the DsDdRequest table references the thread pool. The correct procedure for deleting a thread pool is:

Using isql, update the rules in the DsDdAssignmentRules table so that no requests will be assigned to the thread pool that is to be deleted.

After all requests that are currently assigned to the thread pool have been completed, use isql to set the ThreadLimit in the DsDdThreadPools table to zero.

Wait until all completed requests that were assigned to this thread pool have been garbage collected from the DsDdRequest table. Waiting for 24 hours should ensure this.

Using isql, delete the appropriate row from DsDdThreadPools.

Each rule is defined by a row in the DsDdAssignmentRules table. The following attributes must be specified for each row:

SeqNum determines the order in which a rule is evaluated. It is an integer whose value is greater than or equal to zero. Each rule must have a unique sequence number. Rules are evaluated in order from the lowest sequence number to the highest sequence number. It is recommended that sequence numbers not be created consecutively. So instead of numbering 1, 2, 3, use 100, 200, 300. This will allow new rules to be inserted without having to renumber subsequent rules.

ThreadPoolId is the unique identifier of the thread pool to be assigned if this rule is the first one to evaluate to true. It is an integer whose value must be greater than zero. It must match a value in the ThreadPoolId column in the DsDdThreadPools table. Multiple rules can assign the same ThreadPoolId.

ECSUserId is the user identifier associated with a distribution request. It is a string of up to 24 characters in length. If the user identifier of a distribution request matches this string then this attribute evaluates to true. If the string is set to the reserve word "ANY" then the attribute will always evaluate to true. Note that string comparisons are case sensitive.

Priority is the request priority associated with a distribution request. It is a string that must be set to one of the following six values: "XPRESS", "VHIGH", "HIGH", "NORMAL", "LOW", "ANY". If the priority of a distribution request matches the string then this attribute evaluates to true. If the string is set to the reserve word "ANY" then the attribute will always evaluate to true. Note that string comparisons are case sensitive.

EsdType is the data type associated with a distribution request. It is a string of up to twelve characters in length. It must be set to a valid ESDT name and version number or the reserve words "MULTIPLE" or "ANY". When an ESDT name and version number are specified, the string has the form "Name.Version" (e.g., "MOD021KM.003"). A distribution request will have its EsdType set to "MULTIPLE" if granules from more than one ESDT are being distributed. If the data type of a distribution request matches the string then this attribute evaluates to true. If the string is set to the reserve word "ANY" then the attribute will always evaluate to true. Note that string comparisons are case sensitive.

MediaType is the type of media to be used for a distribution request. It is a string that must be set to one of the following six values: "FtpPush", "FtpPull", "8MM", "CDROM", "DLT", "ANY". If the media type of a distribution request matches the string then this attribute evaluates to true. If the string is set to the reserve word "ANY" then the attribute will always evaluate to true. Note that string comparisons are case sensitive. Also note that "8MM", "CDROM", and "DLT" will currently never appear in a distribution request because media requests are redirected to PDS.

Rules may be added, deleted, or updated at any time by using isql to edit the DsDdAssignmentRules table. If the DDIST server is running when the table is updated, the changes will take effect immediately. That is, any new distribution requests will be allocated to a thread pool using the updated rules. For this reason, the DBA must ensure that rule changes are self-consistent and are typically made within the scope of a single Sybase transaction.

4 Default Thread Pool Tables

Tables 4-1 and 4-2 show the DsDdThreadPools and DsDdAssignmentRules that will be delivered by default with DDIST. These tables implement the current priority-based thread pool scheme with the exception that requests will not be automatically given a thread in a lower priority pool if all threads at their current priority are in use. If a DAAC chooses not to redefine the thread pools then please note the following about the default configuration. The thread limits specified in the default DsDdThreadPools table are suitable for use in the TS1 and TS2 modes. They will likely be too small for use in the OPS mode and should be set to the current DAAC OPS mode values.

Table 4-1 Default DsDdThreadPools Table

ThreadPoolId	ThreadPoolName	ThreadLimit
1	XPRESS	5
2	VHIGH	5
3	HIGH	5
4	NORMAL	5
5	LOW	5
6	DEFAULT	10

Table 4-2 Default DsDdAssignmentRules Table

SeqNum	ECSUserId	Priority	EsdType	MediaType	ThreadPoolId
100	ANY	NORMAL	ANY	ANY	4
200	ANY	LOW	ANY	ANY	5
300	ANY	HIGH	ANY	ANY	3
400	ANY	VHIGH	ANY	ANY	2
500	ANY	XPRESS	ANY	ANY	1

5 Thread Pool Table Examples

Tables 5-1 and 5-2 show an example of how thread pools can be defined for selected individual users. If the userid is not matched then the request is assigned to the DEFAULT pool. Note that the request will be assigned to the first pool where the condition evaluates to true so the ordering of conditions is important.

Table 5-1 DsDdThreadPools Table

ThreadPoolId	ThreadPoolName	ThreadLimit
1	LARC_ING_MGR	3
2	U_MIAMI	3
3	DAAC_STAFF	5
4	DEFAULT	10

Table 5-2 DsDdAssignmentRules Table

SeqNum	ECSUserId	Priority	EsdType	MediaType	ThreadPoolId
--------	-----------	----------	---------	-----------	--------------

100	LarclngMgr	ANY	ANY	ANY	1
200	u.miami	ANY	ANY	ANY	2
300	sharma	ANY	ANY	ANY	3
400	fenichel	ANY	ANY	ANY	3

Tables 5-3 and 5-4 provide a more realistic example of how to configure thread pools to meet GDAAC's current distribution load. It is based on the following distribution strategy:

GDAAC uses two production systems (PDPS and S4PM). Both systems are used to process MODIS data. However, the two systems use different strategies for staging input data. PDPS requests one granule per distribution request and S4PM requests all granules for a given two hour period in a single distribution request. Because MODIS L0 data (MOD000) are large (~6.5 GB per granule), it is desirable to limit the number of granules that are concurrently staged. In order to ensure that L0 data is metered into PDPS, a thread pool will be created for MOD000 staging. An additional thread pool will be created for all other PDPS data types required for production. A single thread pool will be created for all S4PM requests but its thread limit will be kept small to limit the number of concurrent MODIS L0 staging requests.

All distribution requests generated from subscriptions run at HIGH priority. No other distribution requests are permitted to run at HIGH priority. These are all FtpPush requests. One thread pool will be created for each user that uses subscription-based distribution. This enables the number of concurrent transfers to an individual user to be controlled so that a destination site is not overwhelmed. Also, it guarantees that other users are not impacted if network or destination host problems occur with one of the users.

GDAAC uses a DUE, called the Pusher, to resend data to MODAPS and other users when necessary. The Pusher submits acquires to SDSRV using the userid of the original requestor of the data. Pusher requests are always submitted at LOW priority. Because MODAPS is a big user of this utility, a separate thread pool will be created for MODAPS. All other users of the pusher will be grouped into a single additional thread pool.

User requests submitted through the V0GTY are always submitted at NORMAL priority.

In order to throttle hard media throughput, a thread pool will be created for PDS.

Finally, two additional thread pools will be created for user requests for FtpPull and FtpPush distribution.

Remember that rules are evaluated in order by SeqNum. Evaluation stops and the thread pool assignment is made when a rule evaluates to true. If none of the rules evaluate to true, the request is assigned to the DEFAULT thread pool.

Table 5-3 Example GDAAC DsDdThreadPools Table

ThreadPoolId	ThreadPoolName	ThreadLimit
1	PDPS_MOD000	2
2	PDPS_OTHER	5
3	S4PM	2
4	SUB_LARCINGMGR	3
5	SUB_NOAASOAP	3
6	SUB_MODAPS	20
7	SUB_ASTERGDAS	1
8	SUB_CGILI	3
9	SUB_EARMSTRONG	3
10	SUB_INGEST	3
11	SUB_ISIPS	3
12	SUB_LFLYNN	3
13	SUB_LGUMLEY	3
14	SUB_MCSTUSER	5
15	SUB_MOPITT	3
16	SUB_SCHO	3
17	SUB_UMIAMI	5
18	SUB_WINDHOEKANC	3
19	PUSHER_MODAPS	5
20	PUSHER_OTHER	5
21	PDS	10
22	USER_FTPPUSH	10
23	USER_FTPPULL	10
24	DEFAULT	10

Table 5-4 Example GDAAC DsDdAssignmentRules Table

SeqNum	ECSEUserId	Priority	EsdtType	Media Type	ThreadPoolId
100	\$EcDpPrEM	ANY	MOD000.001	ANY	1
200	\$EcDpPrEM	ANY	MODPML0.001	ANY	1
300	\$EcDpPrEM	ANY	ANY	ANY	2
400	s4pmtrops	ANY	ANY	ANY	3
500	s4pmt1ops	ANY	ANY	ANY	3
600	s4pmtfops	ANY	ANY	ANY	3
700	LarclngMgr	HIGH	ANY	FtpPush	4
800	NOAA/SOAP	HIGH	ANY	FtpPush	5
900	MODAPS	HIGH	ANY	FtpPush	6
1000	TERRA_RPROC	HIGH	ANY	FtpPush	6
1100	TERRA_FPROC	HIGH	ANY	FtpPush	6

1200	aster_gdas	HIGH	ANY	FtpPush 7
1300	cgili	HIGH	ANY	FtpPush 8
1400	earmstrong	HIGH	ANY	FtpPush 9
1500	ingest	HIGH	ANY	FtpPush 10
1600	isips	HIGH	ANY	FtpPush 11
1700	lflynn	HIGH	ANY	FtpPush 12
1800	lgumley	HIGH	ANY	FtpPush 13
1900	mcstuser	HIGH	ANY	FtpPush 14
2000	mopitt	HIGH	ANY	FtpPush 15
2100	SCHO	HIGH	ANY	FtpPush 16
2200	u.miami	HIGH	ANY	FtpPush 17
2300	windhoek_anc	HIGH	ANY	FtpPush 18
2400	MODAPS	LOW	ANY	FtpPush 19
2500	TERRA_RPROC	LOW	ANY	FtpPush 19
2600	TERRA_FPROC	LOW	ANY	FtpPush 19
2700	ANY	LOW	ANY	FtpPush 20
2800	\$PDS	ANY	ANY	ANY 21
2900	ANY	ANY	ANY	FtpPush 22
3000	ANY	ANY	ANY	FtpPull 23

6 Monitoring and Troubleshooting Thread Pools

The following sections discuss approaches for troubleshooting common distribution problems.

6.1 Incorrect Thread Pool Assignments

The DDIST GUI has not been modified to show the thread pool associate with each request. Instead, the DDIST server debug log should be used to verify that requests are being assigned to the appropriate thread pools. At debug level 2 or higher, the following two messages will be printed when each new request is assigned to a thread pool.

```
10/26/01 16:31:01: DistRequestS::CalcThreadPoolName. Outgoing cmd:
```

```
exec DsDdTMapReqToThread "1004126988", "cmshared", "HIGH", "L7CPF.002", ""
```

```
10/26/01 16:31:01: DistRequestS::CalcThreadPoolName. Resulting PoolName= :HIGH
```

The first message contains the parameters that are passed to the stored procedure that computes the thread pool assignment. The parameters are Request Id, ECSUserId, Priority, EsdtType, and MediaType. The second message contains the resulting thread pool assignment. Bad assignments typically result from rules that have been incorrectly ordered or because a case-sensitive attribute has been incorrectly typed.

At debug level 2 and higher, the following table is printed every time a distribution request is activated.

PoolName	ThreadLimit	ThreadsAvailable	ActiveCount	PendingCount
PoolName: XPRESS.....	5	5	0	0
PoolName: VHIGH.....	5	5	0	0
PoolName: HIGH.....	5	3	2	0
PoolName: NORMAL.....	5	0	5	21
PoolName: LOW.....	5	0	5	7
PoolName: DEFAULT.....	10	10	0	0

The table shows for each thread pool the thread limit, the number of threads available to process requests, the number of requests that are actively being processed and the number of requests that are pending. Note that this example uses the default thread pool configuration shown in Tables 4-1 and 4-2.

6.2 Remote site is down

This scenario uses the thread pools and assignment rules contained in Tables 5-3 and 5-4. The problem is that the host that contains the directory where user lgumley's FtpPush distribution requests are sent is down. This is noticed by the DAAC Distribution Tech when lgumley requests begin to suspend with errors in the DDIST GUI. If the thread pool configuration has been properly designed (as it has been in Tables 5-3 and 5-4), then this problem will not impact the number of threads that are available for other types of distribution requests. However, this problem could still cause a downstream impact to distribution if not handled properly.

At this point, the operator could let lgumley requests continue to suspend until the site recovers however this needlessly ties up Read Only cache resources on the Archive Servers. This is because each granule is staged into a Read Only cache and then an attempt is made to Ftp the granule to the destination site. When the Ftp fails (because the site is down) the request suspends leaving the file stuck in the Read Only cache until the request is either completed or canceled. If enough lgumley requests are processed while the site is down, a Read Only cache will eventually fill up and block all further distribution from that Archive Server.

A more efficient way to deal with this problem is to have the DBA update the DsDdThreadPools table to set the ThreadLimit on the SUB_LGUMLEY (ThreadPoolId 13) to zero. Within 90 seconds of making this update, DDIST will stop activating requests for this thread pool. Any currently pending or new requests will remain queued. This prevents any more granules from tying up Ready Only cache space while the remote site is down. The only resource that will continue to be tied up are staging disks used for the .met file for each granule. Since these are small, thousands of requests can accumulate in the pending queue without causing problems.

At this point, the operator should periodically resume a suspended request to see if the site is up. Once a request successfully completes the operator should resume the remaining suspended requests and then have the DBA reset the SUB_LGUMLEY thread pool ThreadLimit back to its original value. Within 90 seconds, DDIST will resume activating requests for this thread pool.

6.3 FtpPull Area is Full

This scenario uses the thread pools and assignment rules contained in Tables 5-3 and 5-4. The problem is that the Ftp Pull area on the acg platform is full and there are outstanding FtpPull distribution requests. This is noticed by the DAAC Distribution Tech when Ftp Pull requests begin to suspend with errors in the DDIST GUI. At this point, the operator could let Ftp Pull requests continue to suspend until space becomes available in the Ftp Pull area however this needlessly ties up Read Only cache resources on the Archive Servers. This is because each granule is staged into a Read Only cache and then an attempt is made to Ftp the granule to the Ftp Pull area. When the Ftp fails (because the Pull Area is full) the request suspends leaving the file stuck in the Read Only cache until the request is either completed or canceled. If enough requests are processed while the Pull Area is full, a Read Only cache will eventually fill up and block all further distribution from that Archive Server.

A more efficient way to deal with this problem is to have the DBA update the DsDdThreadPools table to set the ThreadLimit on the USER_FTPPULL (ThreadPoolId 23) to zero. Within 90 seconds of making this update, DDIST will stop activating requests for this thread pool. Any currently pending or new requests will remain queued. This prevents any more granules from tying up Ready Only cache space while the Ftp Pull area is full.

At this point, the Ops Controller should contact the System Engineering staff to correct the Pull Area problem. Once space is available in the Pull Area, the operator should resume the suspended requests and then have the DBA reset the USER_FTPPULL thread pool ThreadLimit back to its original value. Within 90 seconds, DDIST will resume activating requests for this thread pool.

6.4 Increasing the Rate at Which Certain Requests Are Processed

Sometimes it's desirable to increase the rate at which certain distribution requests are processed. This means increasing the number of DDIST worker threads that are available to process the requests. Previously, an operator could accomplish this by changing the priority of a request so that it would be reassigned to a thread pool with available threads. This approach may no longer work if the thread pools have been configured on request attributes other than priority (as in Tables 5-3 and 5-4). In this case, the operator should ask the DBA to temporarily increase the ThreadLimit on the pool which is processing the requests. Once the request backlog has been worked off, the operator should ask the DBA to reset the ThreadLimit to its previous value.

7 Performance and Tuning Guidelines

Following are some simple tuning guidelines.

In most cases, each FtpPush destination site should have its own thread pool.

For each FtpPush destination, the DAAC should determine the number of concurrent file transfers it takes to fully utilize the available network bandwidth. Call this number MaxTransfers.

For subscription based Ftp Push distribution, the thread limit for the associated thread pool should be set to 130% of MaxTransfers (rounded up). This should provide sufficient threads to utilize the available network bandwidth plus allow for one or more threads to be concurrently staging data out of the AMASS cache.

For non-subscription based Ftp Push distribution, the thread limit for the associated thread pool should be set to 200% of MaxTransfers (rounded up). This should provide sufficient threads to utilize the available network bandwidth plus allow for staging of data off of archive tapes.

The total number of threads in DsDdThreadPools (i.e., sum of ThreadLimit for each row) represents the maximum number of threads that can be concurrently be active in DDIST. This total must be less than the number of worker threads configured for DDIST. The default number of worker threads configured for DDIST is 228.

Even though DDIST thread pools can be configured around request attributes other than priority, it's important to remember that STMGT CacheManager thread pools are organized by priority. Thus, one needs to ensure that STMGT thread pools are configured to optimally handle the likely mix of request priorities.

During warm-start, it takes DDIST 0.83 seconds to recover each active or pending request. Thus, with a 2000 request backlog, it will take DDIST approximately 28 minutes to reach the end of start monitoring and begin accepting new requests. Note, however, that it will immediately begin to work off its request backlog as requests are assigned to thread pools.

8 Known Problems, Limitations and Other Strange Behaviors

Following are known problems and limitations related to this capability.

During the testing of this capability, it was found that attempting to resume large numbers of requests at one time (e.g., more than one screen full) would cause the DDIST GUI to get out of synchronization with the DDIST server. This resulted in resume failures. To workaround the problem, simply restart the GUI and resume a smaller number of requests at a time. Typically, resuming one screen full of requests at a time should work.

There is one case when a thread limit has been set to zero in order to temporarily suspend processing in a certain thread pool, where simply setting the thread limit to a nonzero value is not sufficient to restart processing requests that are pending in that thread pool. DDIST will only check for available threads when either a new request arrives or an existing request completes or suspends. If there are requests pending but no new requests arriving or old requests completing in the thread pool then the operator should select a pending request and then suspend/resume it in order to cause DDIST to check for available threads in the pool.

DAAC INSTALLATION INSTRUCTIONS

1. Get TAR Files from CM distribution.

2. If you are untarring the file for multiple modes then untar the file as "cmshared" or change the permissions in the staging area on the files below:

```
cd/<distribution_directory>/<stage_directory>/SGI/CUSTOM/dbms
chmod 744 DSS/EcDsStDbLogin
```

after you untar the file. Ensure that you update ECS Assist when prompted. Please remember that you must be logged in as "root" when you update ECS Assist.

You will get two tar files: One SUN and one IRIX65. Untar both

3. To apply this patch use E.A.S.I to install the following packages:

```
.EcDsDdDISTSRVR.pkg
.EcDsDdDdistGUI.pkg
.EcDsStDatabase.pkg
```

TE_6A.04_DDIST.01 can be loaded on top of Patch_6A.04_DSS.03

4. Check the database patch number for STMGT.

If it is not already at the following versions use

EcsAssist Subsystem Manager to update the STMGT database.

The patch number for STMGT should be 6A85.85